ISYE 6501 HW10

Anonymous for peer grading purposes

2023-01-10

Question 14.1

In this section, we will explore several imputation techniques. The first step, as usual, is to explore the data and perform some basic cleanup. Based on preliminary exploration, we performed the following actions:

1. Missing values are coded as "?", so we read those in as NA for simplicity
2. Remove duplicate rows - 8 total
3. Make the "sample id" our index. Note that some id's are still repeated, so we force uniqueness
4. We note that only V7 (Bare Nuclei) contains missing values. We will focus on filling in the 16 values.

```
library(tidyverse)
library(reshape2)
library(caret)
library(mice)

#Import data
url = paste0('http://archive.ics.uci.edu/ml/machine-learning-databases/',
             'breast-cancer-wisconsin/breast-cancer-wisconsin.data')
data <- read.csv(url, na.strings = '?', header = FALSE)

#Remove duplicates
dups = duplicated(data)
data = unique(data)

#Index on the sample id - force unique ids
rownames(data) <- make.names(data$V1, unique = TRUE)
data = data[,-1]

print(paste(sum(dups), 'duplicates removed'))
```

```
## [1] "8 duplicates removed"
```

```
head(data,4)
```

```
##          V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## X1000025  5  1  1  1  2  1  3  1   1   2
## X1002945  5  4  4  5  7 10  3  2   1   2
## X1015425  3  1  1  1  2  2  3  1   1   2
## X1016277  6  8  8  1  3  4  3  7   1   2
```

```
#How much am I missing?
na_count <-sapply(data, function(x) sum(is.na(x)))
na_count
```

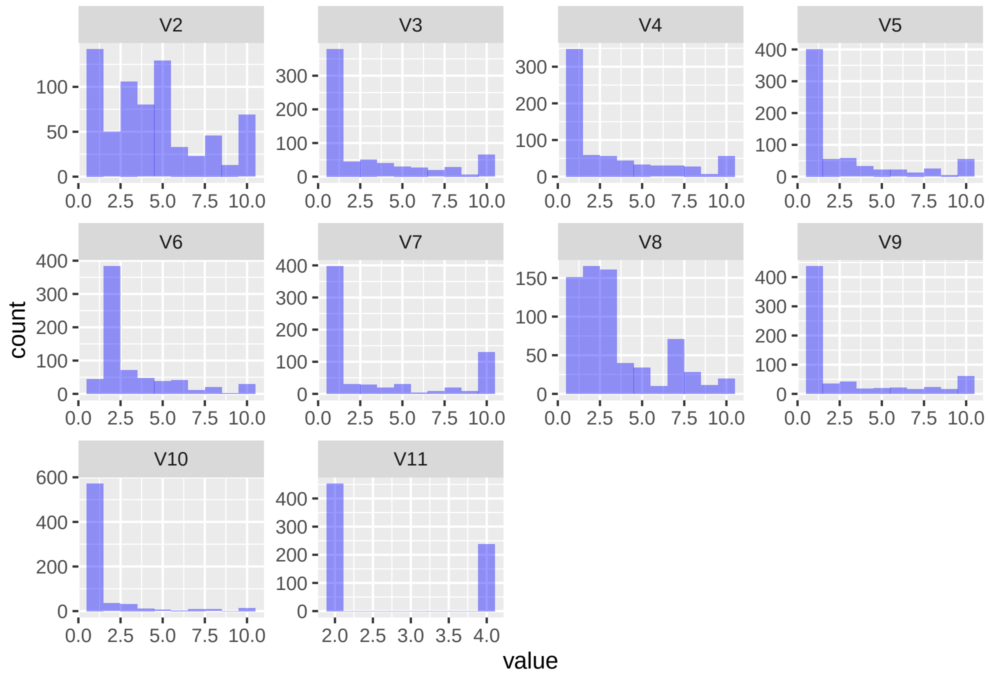
```
##  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11
##   0   0   0   0   0  16   0   0   0   0
```

```r
#Index my missing values for later use
blanks = which(is.na(data), arr.ind=TRUE)
blanks
```

```
##          row col
## X1057013  24   6
## X1096800  41   6
## X1183246 140   6
## X1184840 146   6
## X1193683 159   6
## X1197510 165   6
## X1241232 235   6
## X169356  249   6
## X432809  271   6
## X563649  288   6
## X606140  290   6
## X61634   293   6
## X704168  311   6
## X733639  317   6
## X1238464 406   6
## X1057067 611   6
```

```r
# Melt to plot easily - Lets visualize our data!
melted_data <- melt(data)

ggplot(data=melted_data, aes(x = value)) +
    geom_histogram(bins=10, fill='blue', alpha=0.4) +
    facet_wrap(~variable, scales = "free")
```



NOTE: For all imputation techniques, we use the "mice" package, which can basically handle every imputation method we'd ever need.

1. Mean/Mode Imputation

The easiest method of imputation is Mean imputation (for continuous variables). In this case, our variable is discrete, but taking the mean still applies. Taking the mean of the column yields the expected value, so we simply plug it in in the missing spots.

```
set.seed(10)
impMean <- mice(data,m=1, method='mean')
```

```
##
##  iter imp variable
##   1   1  V7
##   2   1  V7
##   3   1  V7
##   4   1  V7
##   5   1  V7
```

```
data_impMean = complete(impMean,1)

print(data_impMean[blanks])
```

```
##  [1] 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778
##  [9] 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778 3.537778
```

2. Regression Imputation

Another method of imputation is to build a model that predicts our missing feature based on the available data. Using the model, we can then fill in the missing values with the model predictions! This often results in better estimates for our missing values, but may cause overfitting.

```
set.seed(10)
impReg <- mice(data,m=1, method='norm.predict')
```

```
##
##  iter imp variable
##   1   1  V7
##   2   1  V7
##   3   1  V7
##   4   1  V7
##   5   1  V7
```

```
data_impReg = complete(impReg,1)

print(data_impReg[blanks])
```

```
##  [1] 7.136237 3.492670 1.198277 1.557537 1.247025 1.456574 1.928802 1.407641
##  [9] 1.606471 6.315448 1.210096 1.011211 2.062107 1.407641 1.198277 1.049666
```

3. Perturbed Regression Imputation

Yet another method of imputation is to, once again, build a model that predicts our missing feature based on the available data. This time however, we add an error term in order to add some variance to the predictions. The mice package accomplishes this via the norm.nob method, which uses the spread around the fitted linear regression line to create the new values.

```
set.seed(10)
impRegPert <- mice(data,m=1, method='norm.nob')
```

```
##
```

```
##  iter imp variable
##   1   1  V7
##   2   1  V7
##   3   1  V7
##   4   1  V7
##   5   1  V7
```

```r
data_impRegPert = complete(impRegPert,1)

print(data_impRegPert[blanks])
```

```
##  [1]  7.7102475  6.1137344 -0.5122707 -0.4611775 -0.5903428  0.3488799
##  [7] -0.6163562  1.4902082  1.1664619  5.0186582  5.4387792  1.4676961
## [13]  3.4918269  2.5059870  0.9542487 -0.3772533
```

4. Classification Model Comparison - Missing Data

In this section we will explore the performance of the same classification model, just using the different imputed values for V7. We use a Random Forest classification model because it is easy to implement and it works (pretty much) right out of the box. In addition, we are only interested in comparing the results of each imputation technique to each other, so explanatory power is not essential. Key Results are summarized here:

1. Mean Imputation:
    - 2.89% OOB error
2. Regression Imputation:
    - 3.04% OOB error
3. Perturbed Regression:
    - 2.75% OOB error
4. Removed Missing Values:
    - 2.81 OOB error

```r
set.seed(10)

data_naRM <- data[complete.cases(data),]

cv <- trainControl(method = "cv",  number = 5)

# CV random forest
mMean <- train(as.factor(V11) ~ ., data = data_impMean,
               method = "rf", trControl = cv, importance = TRUE)
mReg <- train(as.factor(V11) ~ ., data = data_impReg,
              method = "rf", trControl = cv, importance = TRUE)
mRegPert <- train(as.factor(V11) ~ ., data = data_impRegPert,
                  method = "rf", trControl = cv, importance = TRUE)
mRemoved <- train(as.factor(V11) ~ ., data = data_naRM,
                  method = "rf", trControl = cv, importance = TRUE)

Mean_model = mMean$finalModel
Reg_model = mReg$finalModel
RegPert_model = mRegPert$finalModel
naRm_model = mRemoved$finalModel


print(Mean_model)
```

```
##
```

```
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.89%
## Confusion matrix:
##     2   4 class.error
## 2 440  13  0.02869757
## 4   7 231  0.02941176
```

```
print(Reg_model)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.04%
## Confusion matrix:
##     2   4 class.error
## 2 440  13  0.02869757
## 4   8 230  0.03361345
```

```
print(RegPert_model)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.75%
## Confusion matrix:
##     2   4 class.error
## 2 440  13  0.02869757
## 4   6 232  0.02521008
```

```
print(naRm_model)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.81%
## Confusion matrix:
##     2   4 class.error
## 2 428  11  0.02505695
## 4   8 228  0.03389831
```

Imputation Summary

Imputation can be a very useful tool for dealing with missing data. In this particular case, all 4 methods were relatively close in performance with a range of 2.75-3.04% OOB error . In fact, 2 of the 3 imputation methods actually increased the error of the model (vs removing the data)! This may be due to the fact that only 16 features from the same variable are missing, so we aren't losing too much information with deletion. Nonetheless, since we are analyzing breast cancer, even 1 incorrect classification could have huge implications, so every bit of performance matters. In practice, we would spend a lot more time and effort on the model building process, but for the purposes of this exercise, we pick the perturbed imputation method for a 5-fold cv error estimate of 2.75%.

Question 15.1

I currently work at a B2B technology company specializing in conversational analytics. From a a business perspective, optimization would be quite beneficial for management of call center agents. Most of our clients operate call centers from a few dozens to a few thousands of employees! As a provider of conversational analytics solutions, we could develop a system to maximize service level while minimizing wait times and operational cost. Assuming we were contracted to do this as a professional service, I would aim to collect the following data:

1. Total agents available on a given day
2. Regulatory and employer requirements
3. Average call duration
4. Average pre- and post-call handling time
5. Hours of operation
6. Estimated call volume per day of the week
7. Upcoming promotions and release dates
8. Seasonality parameters
9. Etc.